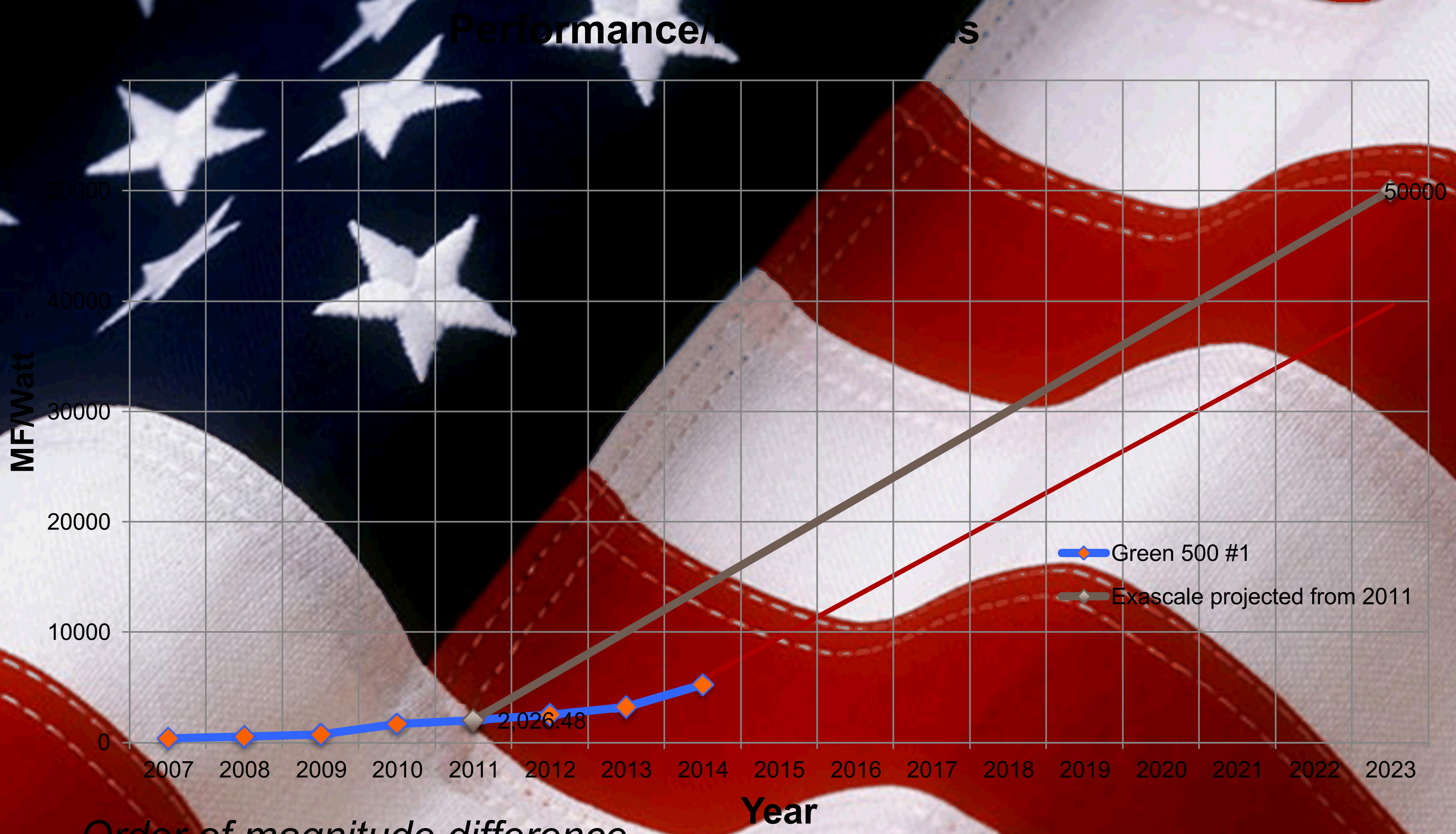
SAND2015-6774C

Monitoring Large-Scale HPC Systems Workshop

Power [illegible]surement and Control
James [illegible]
Sandia [illegible]aboratories
http://powerapi.sandia.gov

Power - Historic Trends

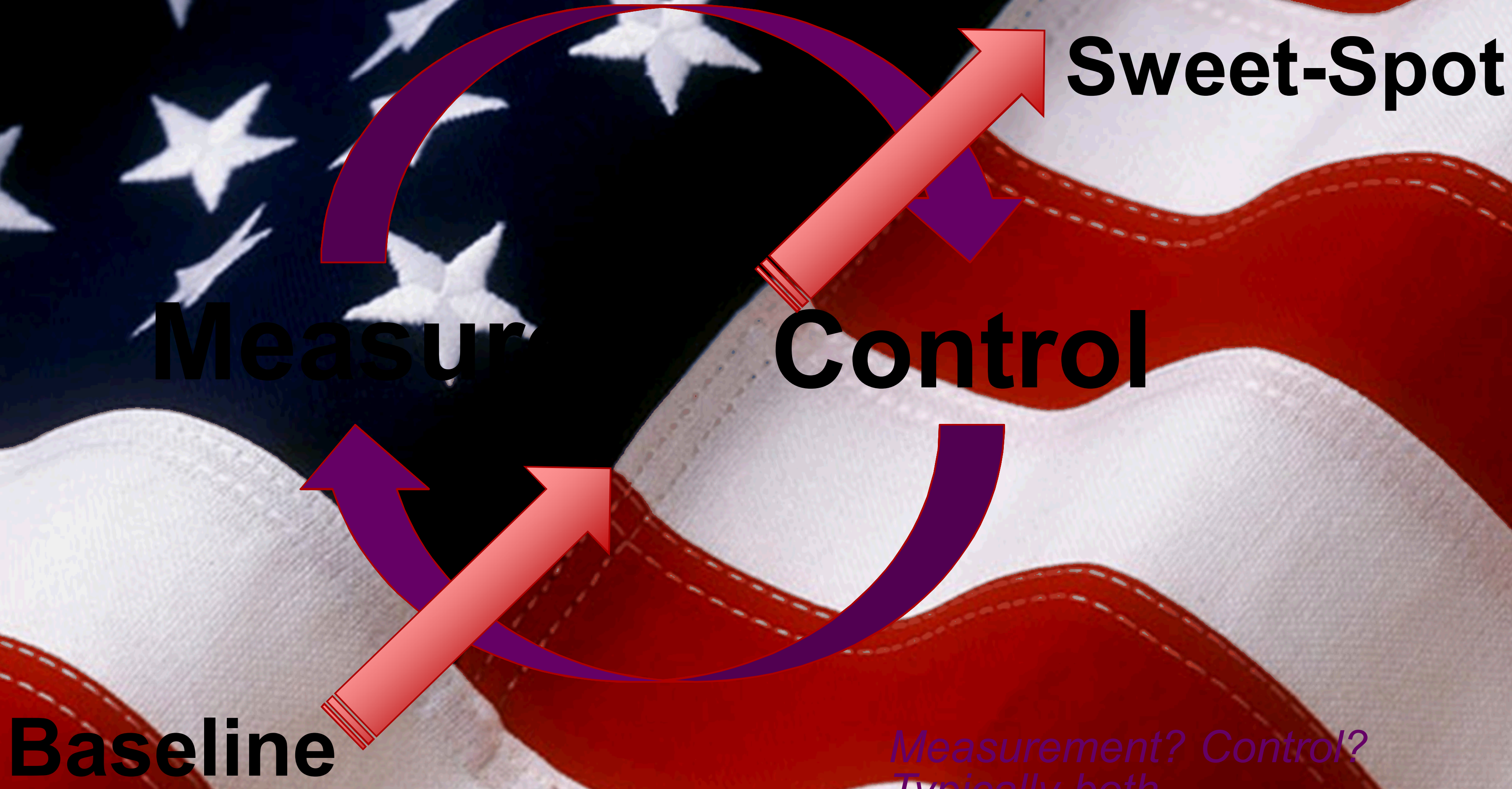
Motivation



Order of magnitude difference in our application efficiency

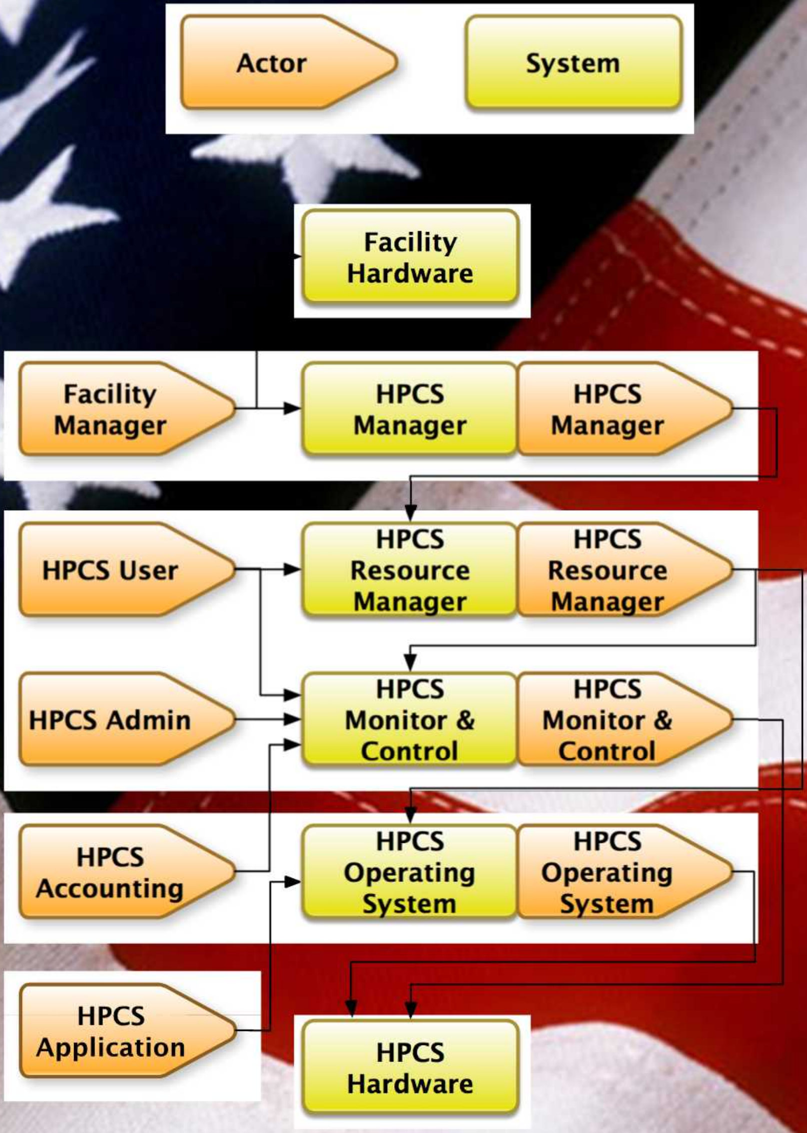
Falls short by 200 PetaFlops

What do most Use Cases have in Common?



Measurement? Control? Typically both.

Driven by Use Cases



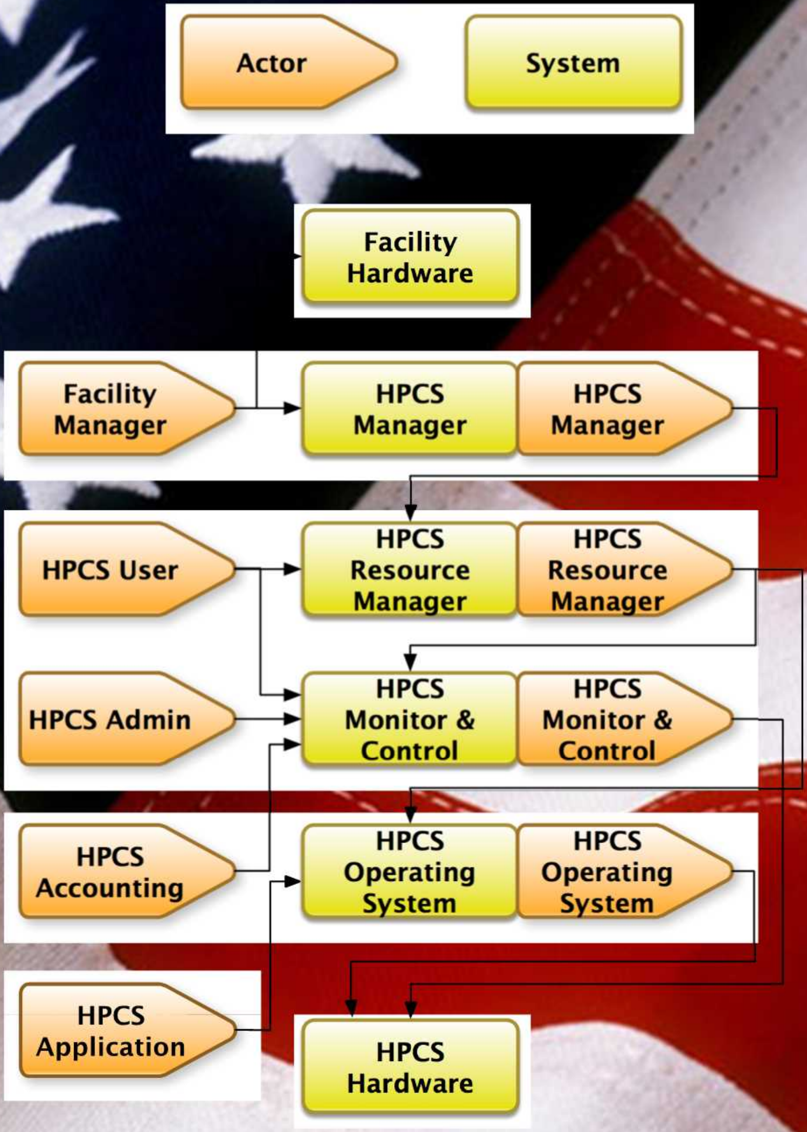
Example Use Cases

- [illegible] Aware Scheduling
  - Ask 12 people get 12 different [illegible]
    - At least the answers [illegible]ed
- Trinity Use Case

1. Run Application in de[illegible]e
2. MEASURE Power a[illegible]y
   - Point in time [illegible]lows us to produce Application Power Profile
   - Energy meas[illegible]ws us to establish baseline
   - What are the i[illegible]ions of this MEASUREMENT step?
3. Run Application [illegible]th adjusted frequency (for example)
   - CONTROL part of loop
   - What are the implications of enabling this kind of CONTROL?
4. Goto #2

Audience Participation

What Role/System combinations have we exercised?



Questions?

Sandia National Laboratories

Exceptional service in the national interest

Backup Slides

System Description

PWR_ObjType

```
typedef enum {
        PWR_OBJ_PLATFORM,
        PWR_OBJ_CABINET,
        PWR_OBJ_CHASSIS,
        PWR_OBJ_BOARD,
        PWR_OBJ_NODE,
        PWR_OBJ_SOCKET,
        PWR_OBJ_CORE,
        PWR_OBJ_POWER_PLANE,
        PWR_OBJ_MEM,
        PWR_OBJ_NIC,
        PWR_OBJ_INVALID
} PWR_ObjType;
```

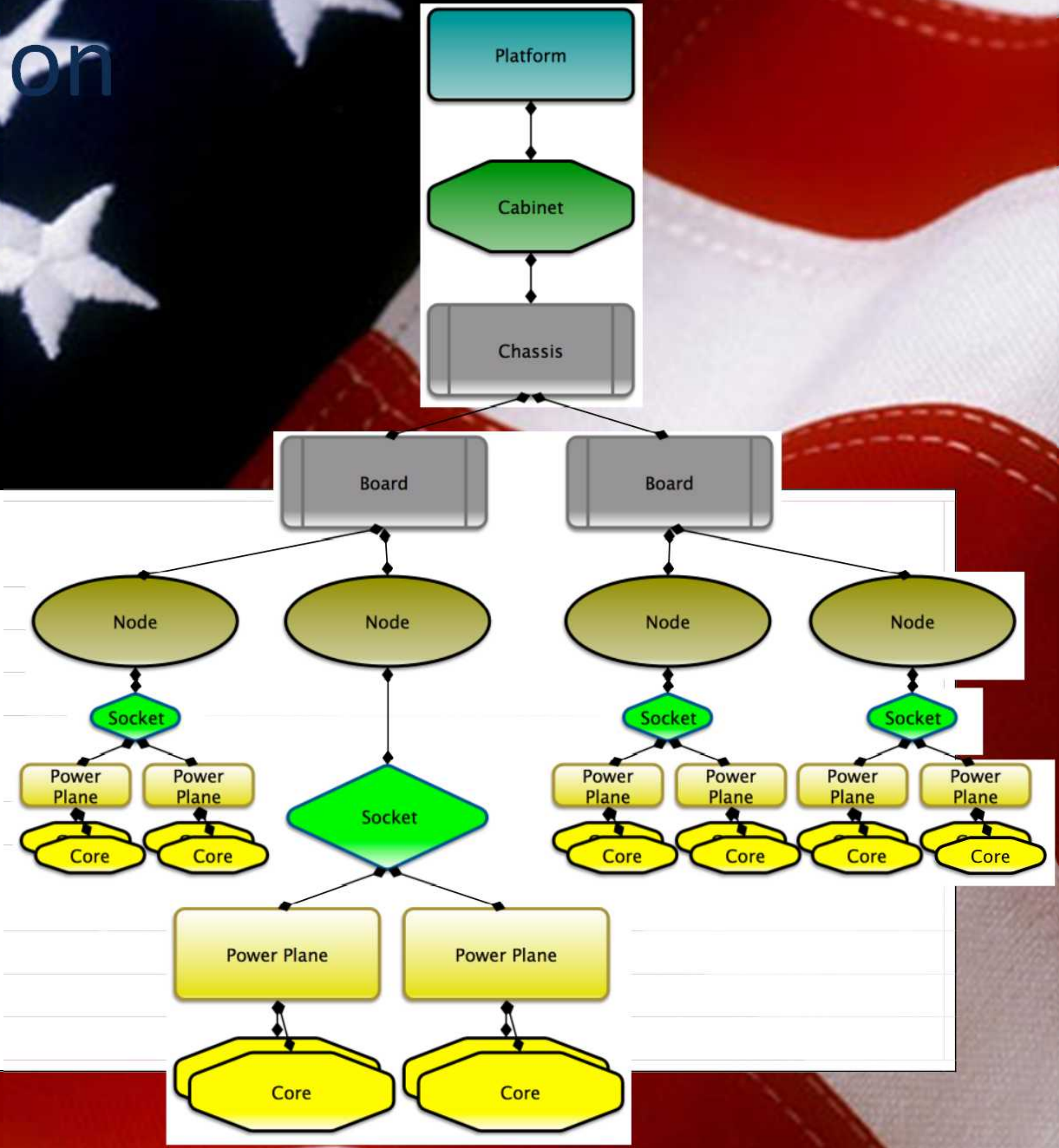
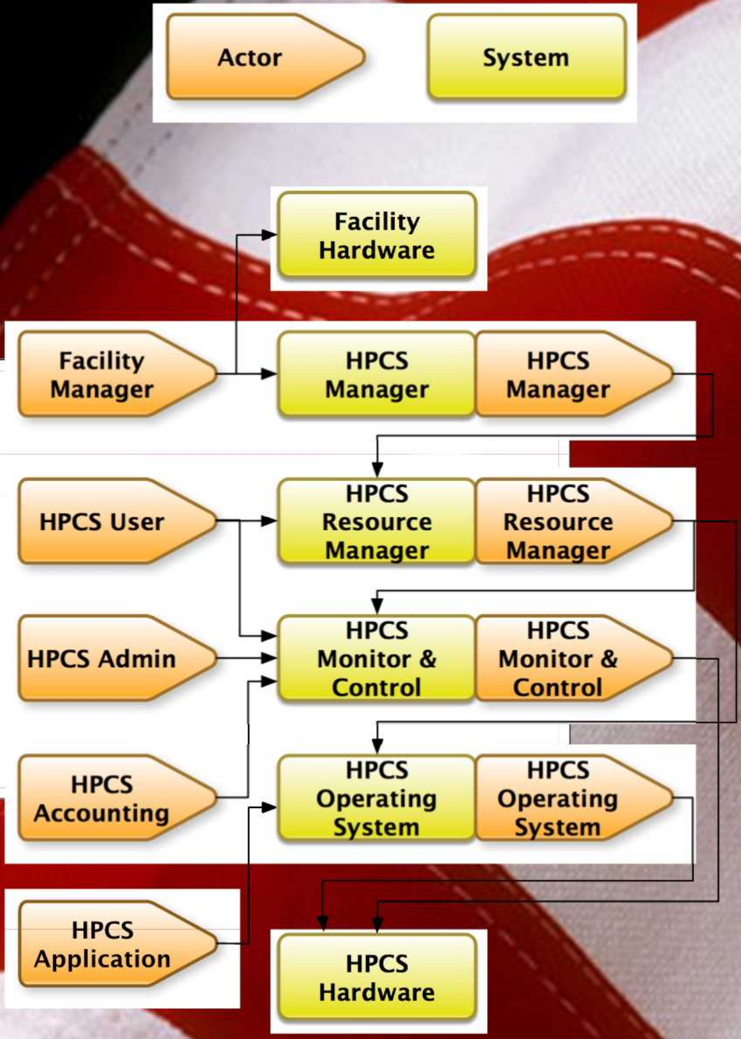


Figure 2.1: Hierarchical Depiction of System Objects

Roles

PWR_Role

```
typedef enum {
            PWR_ROLE_APP, /* Application */
            PWR_ROLE_MC, /* Monitor and Control */
            PWR_ROLE_OS, /* Operating System */
            PWR_ROLE_USER, /* User */
            PWR_ROLE_RM, /* Resource Manager */
            PWR_ROLE_ADMIN, /* Administrator */
            PWR_ROLE_MGR, /* HPCS Manager */
            PWR_ROLE_ACC /* Accounting */
} PWR_Role;
```

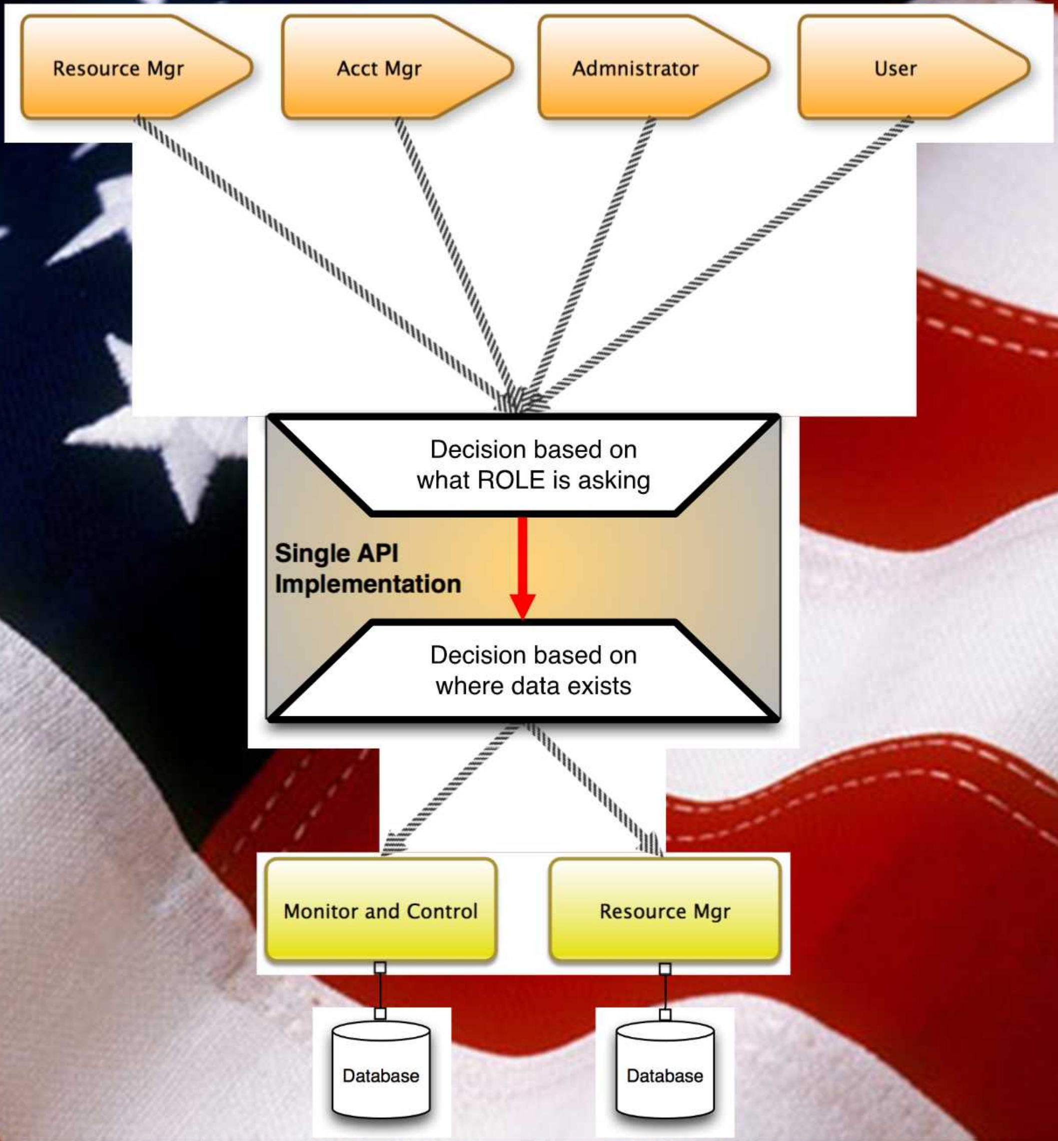


Foundation: Measurement and Control

PWR_AttrName

```
typedef enum {
     PWR_ATTR_PSTATE = 0, /* uint64_t */
     PWR_ATTR_CSTATE, /* uint64_t */
     PWR_ATTR_CSTATE_LIMIT, /* uint64_t */
     PWR_ATTR_SSTATE, /* uint64_t */
     PWR_ATTR_POWER, /* double, Watts */
     PWR_ATTR_CURRENT, /* double, Amps */
     PWR_ATTR_VOLTAGE, /* double, Voltage */
     PWR_ATTR_MAX_POWER, /* double, Watts */
     PWR_ATTR_MIN_POWER, /* double, Watts */
     PWR_ATTR_FREQ, /* double, Hz */
     PWR_ATTR_ENERGY, /* double, Joules */
     PWR_ATTR_TEMP, /* double, Celsius */
     PWR_ATTR_OS_ID, /* uint64_t */
     PWR_ATTR_NUM_ATTRS,
     PWR_ATTR_INVALID = PWR_ATTR_NUM_ATTRS,
} PWR_AttrName;
```

Higher Level Interfaces


Resource Mgr
Acct Mgr
Admnistrator
User
Decision based on what ROLE is asking
Single API Implementation
Decision based on where data exists
Monitor and Control
Resource Mgr
Database
Database